

Convolutional Neural Network (ConvNet/CNN)

Deep Learning (DSE316/616)

Vinod K Kurmi
Assistant Professor, DSE

Indian Institute of Science Education and Research Bhopal

Sep 15, 2022

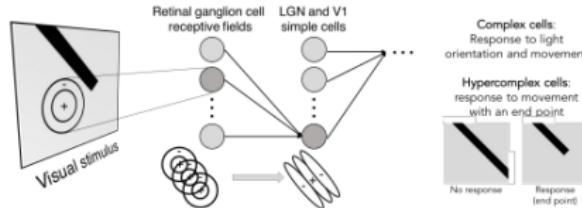
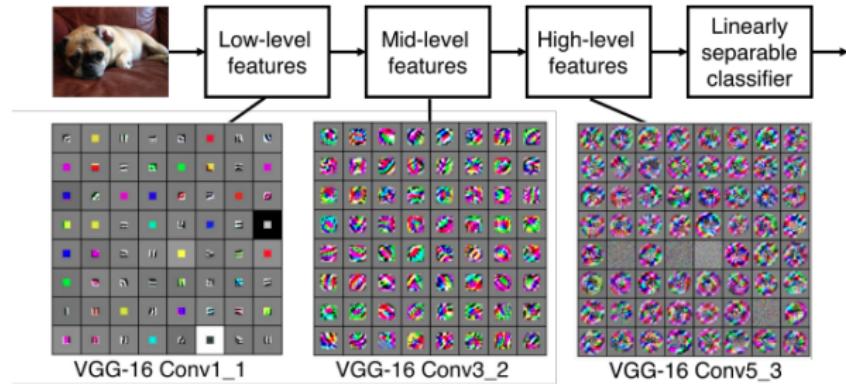


Disclaimer

- Much of the material and slides for this lecture were borrowed from
 - Bernhard Schölkopf's MLSS 2017 lecture,
 - Tommi Jaakkola's 6.867 class,
 - CMP784: Deep Learning Fall 2021 Erkut Erdem Hacettepe University
 - Fei-Fei Li, Andrej Karpathy and Justin Johnson's CS231n class
 - Hongsheng Li's ELEG5491 class
 - Tsz-Chiu Au slides
 - Mitesh Khapra Class notes

Previous class: CNN

Preview



Convolution operation is element wise multiply and add

1	0	1
0	1	0
1	0	1

Filter / Kernel

1 $\times 1$	1 $\times 0$	1 $\times 1$	0	0
0 $\times 0$	1 $\times 1$	1 $\times 0$	1	0
0 $\times 1$	0 $\times 0$	1 $\times 1$	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved Feature

Producing Feature Maps



Original



Sharpen



Edge Detect



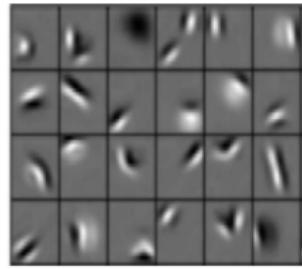
“Strong” Edge Detect

Simple Kernels / Filters

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

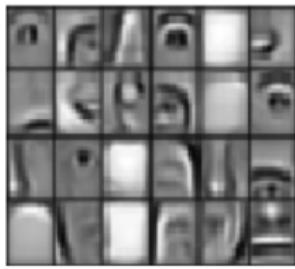
Key idea: learn hierarchy of features directly from the data
(rather than hand-engineering them)

Low level features



Edges, dark spots

Mid level features



Eyes, ears, nose

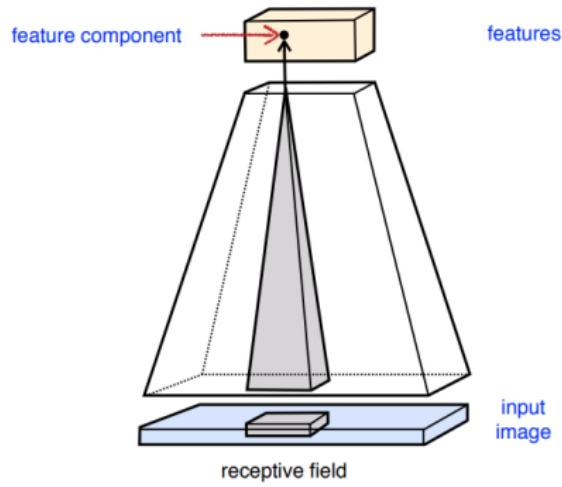
High level features



Facial structure

Convolutional layers

- Local receptive field
- Each column of hidden units looks at a different input patch



LeNet-5

- Gradient Based Learning Applied To Document Recognition - Y. Lecun, L. Bottou, Y. Bengio, P. Haffner; 1998
- Helped establish how we use CNNs today
- Replaced manual feature extraction

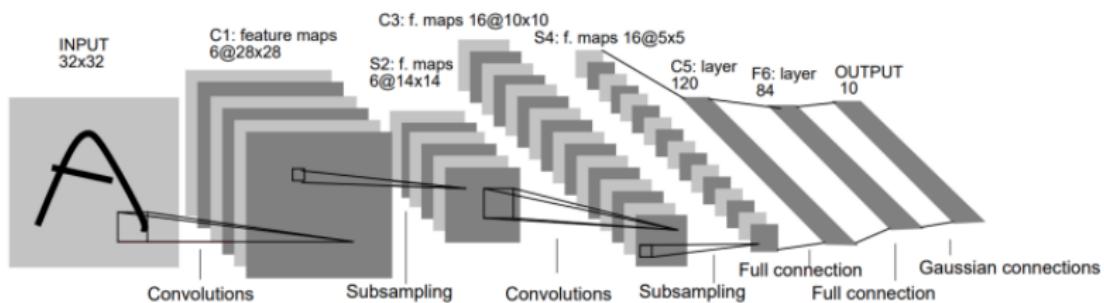
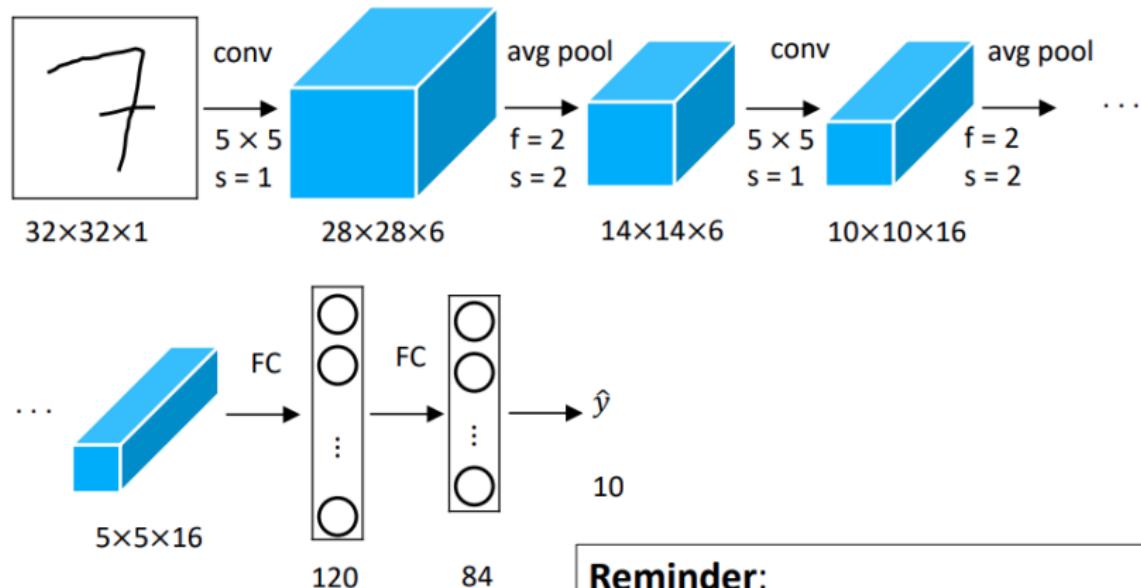


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

LeNet-5



Reminder:

Output size = $(N+2P-F)/\text{stride} + 1$

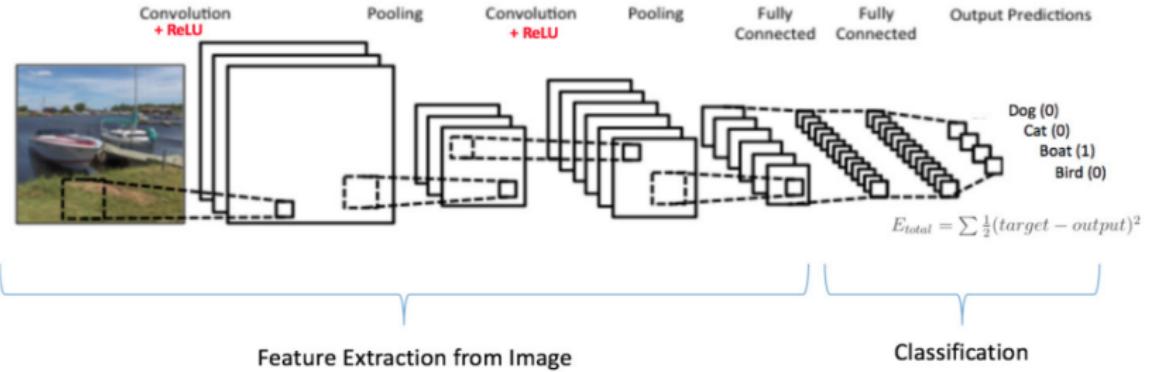
This slide is taken from Andrew Ng

[LeCun et al., 1998]

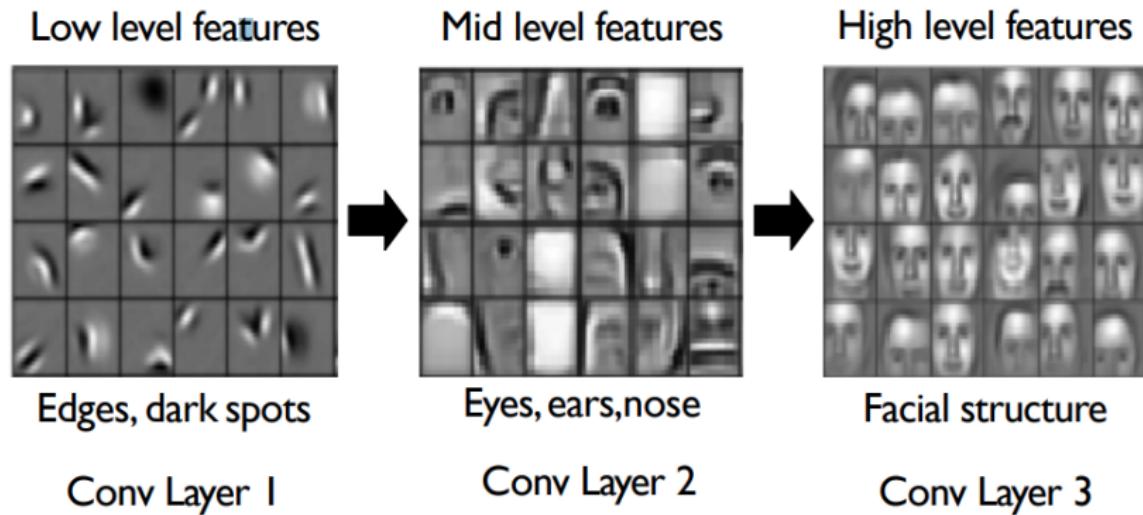
LeNet-5

- Only 60K parameters
- As we go deeper in the network: N_h, N_w, N_c
- General structure: conv->pool->conv->pool->FC->FC->output
- Different filters look at different channels
- Sigmoid and Tanh nonlinearity

An image classification CNN

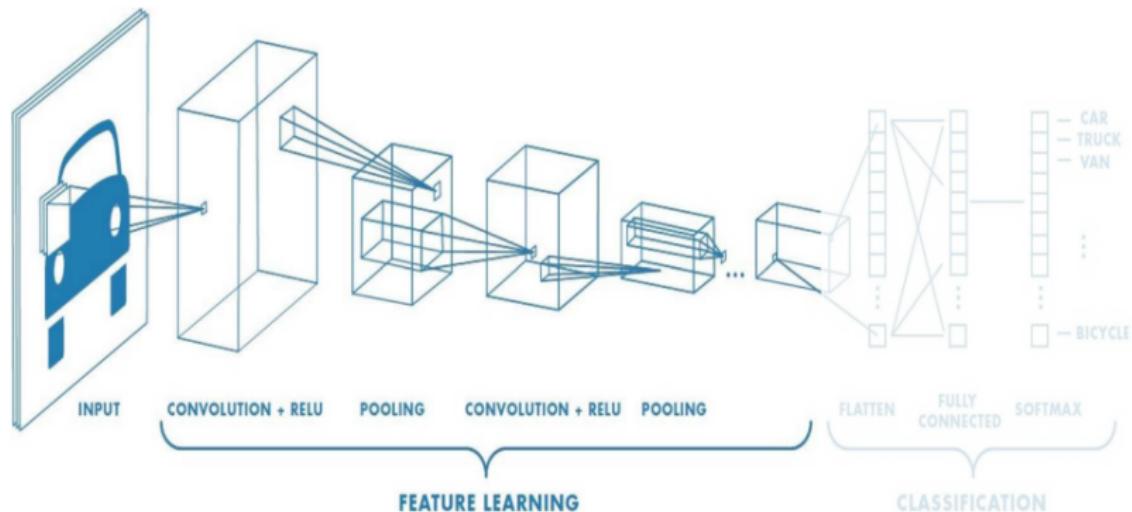


Representation Learning in Deep CNNs



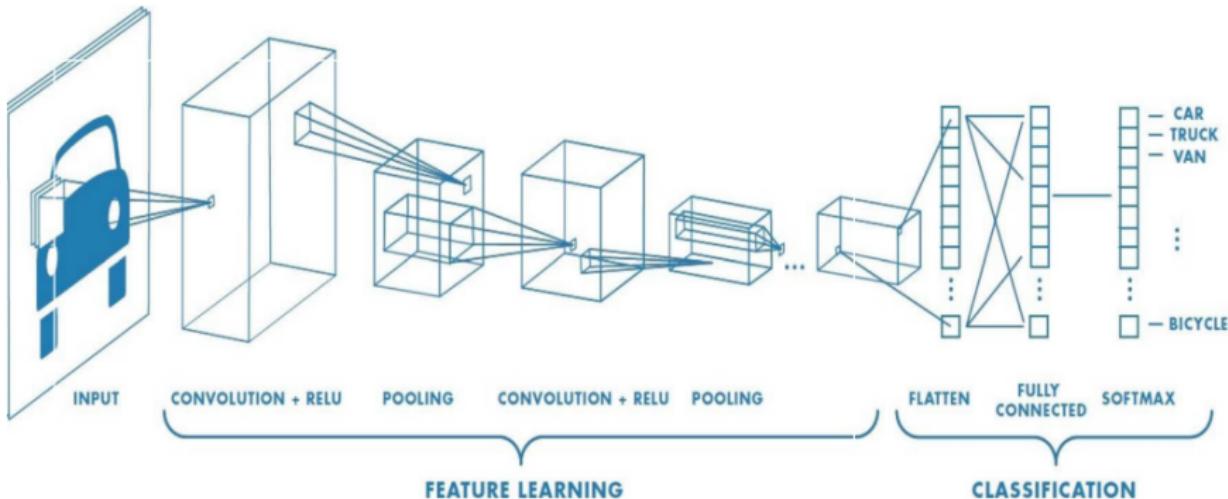
Lee+ ICML 2009

CNNs for Classification: Feature Learning



- Learn features in input image through convolution
- Introduce non-linearity through activation function (real-world data is non-linear!)
- Reduce dimensionality and preserve spatial invariance with pooling

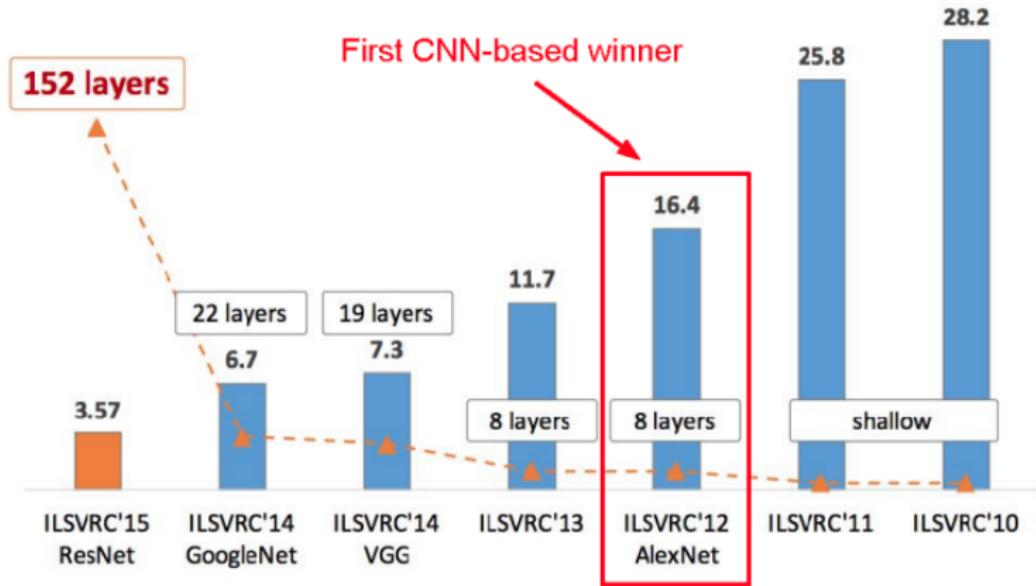
CNNs for Classification: Class Probabilities



$$\text{softmax}(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

- CONV and POOL layers output high-level features of input
- Fully connected layer uses these features for classifying input image
- Express output as probability of image belonging to a particular class

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



AlexNet

- ImageNet Classification with Deep Convolutional Neural Networks - Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton; 2012
- Facilitated by GPUs, highly optimized convolution implementation and large datasets (ImageNet)
- One of the largest CNNs to date
- Has 60 Million parameter compared to 60k parameter of LeNet-5

AlexNet

Architecture

CONV1

MAX POOL1

NORM1

CONV2

MAX POOL2

NORM2

CONV3

CONV4

CONV5

Max POOL3

FC6

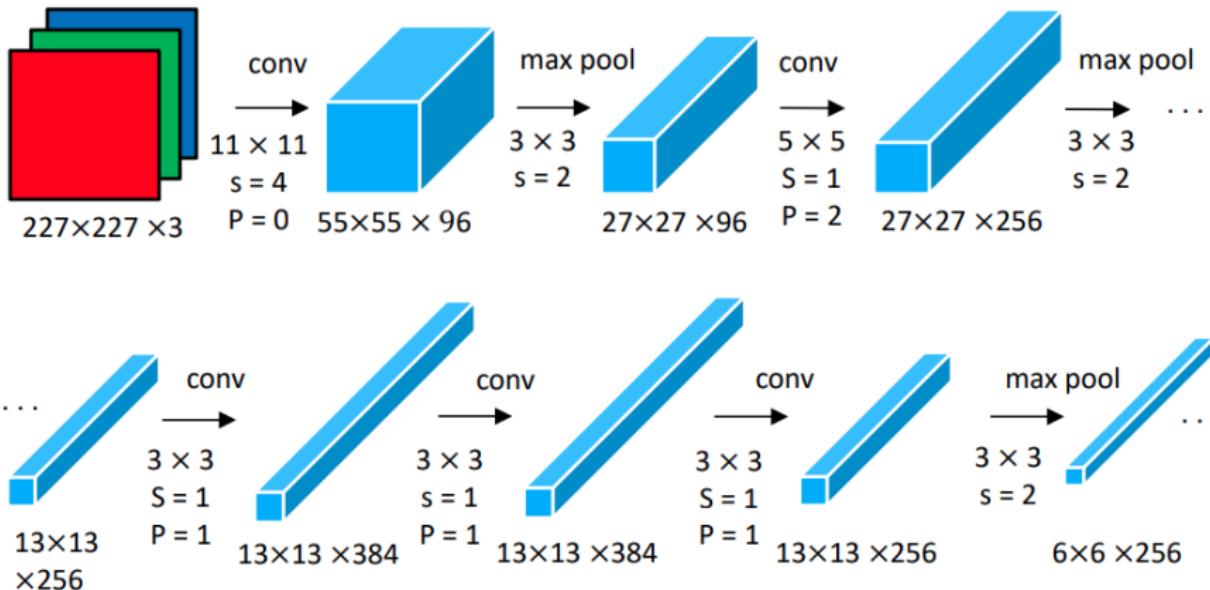
FC7

FC8

AlexNet

- Input: 227x227x3 images (224x224 before padding)
- First layer: 96 11x11 filters applied at stride 4
- **Output volume size?**
$$(N-F)/s+1 = (227-11)/4+1 = 55 \rightarrow [55x55x96]$$
- **Number of parameters in this layer?**
$$(11*11*3)*96 = 35K$$

AlexNet



This slide is taken from Andrew Ng

[Krizhevsky et al., 2012]

AlexNet

Details/Retrospectives

- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- 7 CNN ensemble

AlexNet

- Trained on GTX 580 GPU with only 3 GB of memory.
- Network spread across 2 GPUs, half the neurons (feature maps) on each GPU.
- CONV1, CONV2, CONV4, CONV5: Connections only with feature maps on same GPU.
- CONV3, FC6, FC7, FC8: Connections with all feature maps in preceding layer, communication across GPUs.

AlexNet

Case Study: AlexNet

[Krizhevsky et al. 2012]

Full (simplified) AlexNet architecture:

[227x227x3] INPUT

[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0

[27x27x96] MAX POOL1: 3x3 filters at stride 2

[27x27x96] NORM1: Normalization layer

[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2

[13x13x256] MAX POOL2: 3x3 filters at stride 2

[13x13x256] NORM2: Normalization layer

[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1

[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1

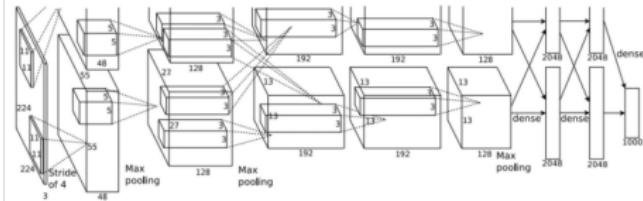
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1

[6x6x256] MAX POOL3: 3x3 filters at stride 2

[4096] FC6: 4096 neurons

[4096] FC7: 4096 neurons

[1000] FC8: 1000 neurons (class scores)

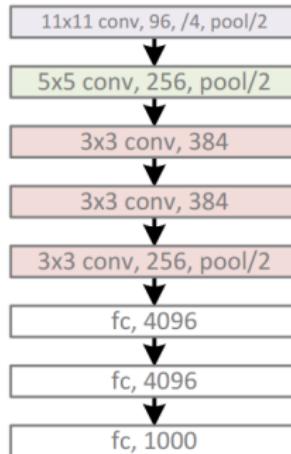


Details/Retrospectives:

- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%

Revolution of Depth

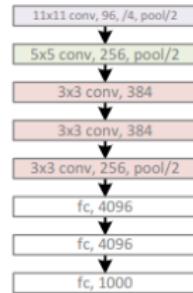
AlexNet, 8 layers
(ILSVRC 2012)



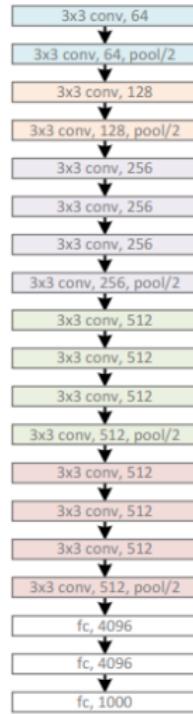
- 5 convolutional layers
- 3 fully connected layers
- ReLU
- End-to-end (no pre-training)
- Data augmentation

Revolution of Depth

AlexNet, 8 layers
(ILSVRC 2012)



VGG, 19 layers
(ILSVRC 2014)



- Very deep
- Simply deep

INPUT: [224x224x3] memory: $224 \times 224 \times 3 = 150\text{K}$ params: 0 (not counting biases)
 CONV3-64: [224x224x64] memory: $224 \times 224 \times 64 = 3.2\text{M}$ params: $(3 \times 3 \times 3) \times 64 = 1,728$
 CONV3-64: [224x224x64] memory: $224 \times 224 \times 64 = 3.2\text{M}$ params: $(3 \times 3 \times 64) \times 64 = 36,864$
 POOL2: [112x112x64] memory: $112 \times 112 \times 64 = 800\text{K}$ params: 0
 CONV3-128: [112x112x128] memory: $112 \times 112 \times 128 = 1.6\text{M}$ params: $(3 \times 3 \times 64) \times 128 = 73,728$
 CONV3-128: [112x112x128] memory: $112 \times 112 \times 128 = 1.6\text{M}$ params: $(3 \times 3 \times 128) \times 128 = 147,456$
 POOL2: [56x56x128] memory: $56 \times 56 \times 128 = 400\text{K}$ params: 0
 CONV3-256: [56x56x256] memory: $56 \times 56 \times 256 = 800\text{K}$ params: $(3 \times 3 \times 128) \times 256 = 294,912$
 CONV3-256: [56x56x256] memory: $56 \times 56 \times 256 = 800\text{K}$ params: $(3 \times 3 \times 256) \times 256 = 589,824$
 CONV3-256: [56x56x256] memory: $56 \times 56 \times 256 = 800\text{K}$ params: $(3 \times 3 \times 256) \times 256 = 589,824$
 POOL2: [28x28x256] memory: $28 \times 28 \times 256 = 200\text{K}$ params: 0
 CONV3-512: [28x28x512] memory: $28 \times 28 \times 512 = 400\text{K}$ params: $(3 \times 3 \times 256) \times 512 = 1,179,648$
 CONV3-512: [28x28x512] memory: $28 \times 28 \times 512 = 400\text{K}$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$
 CONV3-512: [28x28x512] memory: $28 \times 28 \times 512 = 400\text{K}$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$
 POOL2: [14x14x512] memory: $14 \times 14 \times 512 = 100\text{K}$ params: 0
 CONV3-512: [14x14x512] memory: $14 \times 14 \times 512 = 100\text{K}$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$
 CONV3-512: [14x14x512] memory: $14 \times 14 \times 512 = 100\text{K}$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$
 CONV3-512: [14x14x512] memory: $14 \times 14 \times 512 = 100\text{K}$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$
 POOL2: [7x7x512] memory: $7 \times 7 \times 512 = 25\text{K}$ params: 0
 FC: [1x1x4096] memory: 4096 params: $7 \times 7 \times 512 \times 4096 = 102,760,448$
 FC: [1x1x4096] memory: 4096 params: $4096 \times 4096 = 16,777,216$
 FC: [1x1x1000] memory: 1000 params: $4096 \times 1000 = 4,096,000$

TOTAL memory: $24\text{M} * 4 \text{ bytes} \approx 93\text{MB} / \text{image}$

(only forward! \sim^2 for bwd)

TOTAL params: 138M parameters

ConvNet Configuration		
B	C	D
13 weight layers	16 weight layers	16 weight layers
put (224 x 224 RGB image)		
conv3-64	conv3-64	conv3-64
conv3-64	conv3-64	conv3-64
maxpool		
conv3-128	conv3-128	conv3-128
conv3-128	conv3-128	conv3-128
maxpool		
conv3-256	conv3-256	conv3-256
conv3-256	conv3-256	conv3-256
conv1-256	conv3-256	conv3-256
maxpool		
conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512
conv1-512	conv3-512	conv3-512
maxpool		
conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512
conv1-512	conv3-512	conv3-512
maxpool		
FC-4096		
FC-4096		
FC-1000		
soft-max		

VGG-16 Net

INPUT: [224x224x3] memory: $224 \times 224 \times 3 = 150K$ params: 0 (not counting biases)
 CONV3-64: [224x224x64] memory: $224 \times 224 \times 64 = 3.2M$ params: $(3 \times 3 \times 3) \times 64 = 1,728$
 CONV3-64: [224x224x64] memory: $224 \times 224 \times 64 = 3.2M$ params: $(3 \times 3 \times 64) \times 64 = 36,864$
 POOL2: [112x112x64] memory: $112 \times 112 \times 64 = 800K$ params: 0
 CONV3-128: [112x112x128] memory: $112 \times 112 \times 128 = 1.6M$ params: $(3 \times 3 \times 64) \times 128 = 73,728$
 CONV3-128: [112x112x128] memory: $112 \times 112 \times 128 = 1.6M$ params: $(3 \times 3 \times 128) \times 128 = 147,456$
 POOL2: [56x56x128] memory: $56 \times 56 \times 128 = 400K$ params: 0
 CONV3-256: [56x56x256] memory: $56 \times 56 \times 256 = 800K$ params: $(3 \times 3 \times 128) \times 256 = 294,912$
 CONV3-256: [56x56x256] memory: $56 \times 56 \times 256 = 800K$ params: $(3 \times 3 \times 256) \times 256 = 589,824$
 CONV3-256: [56x56x256] memory: $56 \times 56 \times 256 = 800K$ params: $(3 \times 3 \times 256) \times 256 = 589,824$
 POOL2: [28x28x256] memory: $28 \times 28 \times 256 = 200K$ params: 0
 CONV3-512: [28x28x512] memory: $28 \times 28 \times 512 = 400K$ params: $(3 \times 3 \times 256) \times 512 = 1,179,648$
 CONV3-512: [28x28x512] memory: $28 \times 28 \times 512 = 400K$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$
 CONV3-512: [28x28x512] memory: $28 \times 28 \times 512 = 400K$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$
 POOL2: [14x14x512] memory: $14 \times 14 \times 512 = 100K$ params: 0
 CONV3-512: [14x14x512] memory: $14 \times 14 \times 512 = 100K$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$
 CONV3-512: [14x14x512] memory: $14 \times 14 \times 512 = 100K$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$
 CONV3-512: [14x14x512] memory: $14 \times 14 \times 512 = 100K$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$
 POOL2: [7x7x512] memory: $7 \times 7 \times 512 = 25K$ params: 0
 FC: [1x1x4096] memory: 4096 params: $7 \times 7 \times 512 \times 4096 = 102,760,448$
 FC: [1x1x4096] memory: 4096 params: $4096 \times 4096 = 16,777,216$
 FC: [1x1x1000] memory: 1000 params: $4096 \times 1000 = 4,096,000$

TOTAL memory: $24M * 4 \text{ bytes} \approx 93\text{MB} / \text{image}$

(only forward! \sim^2 for bwd)

TOTAL params: 138M parameters

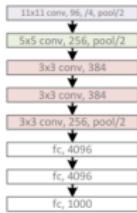
Note:

Most memory is in early CONV

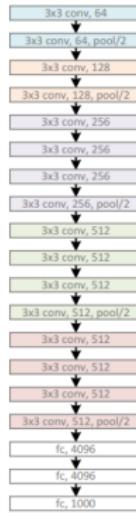
Most params are in late FC

VGG-16 Net

AlexNet, 8 layers
(ILSVRC 2012)

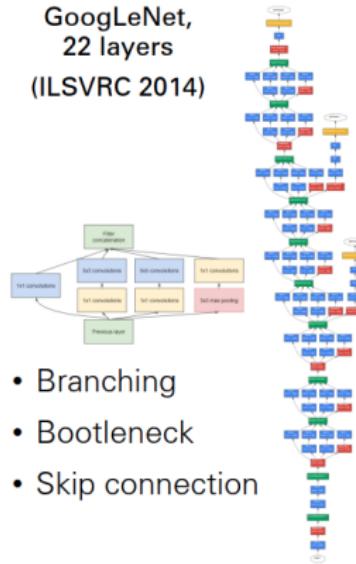


VGG, 19 layers
(ILSVRC 2014)



- Very deep
- Simply deep

GoogLeNet, 22 layers
(ILSVRC 2014)

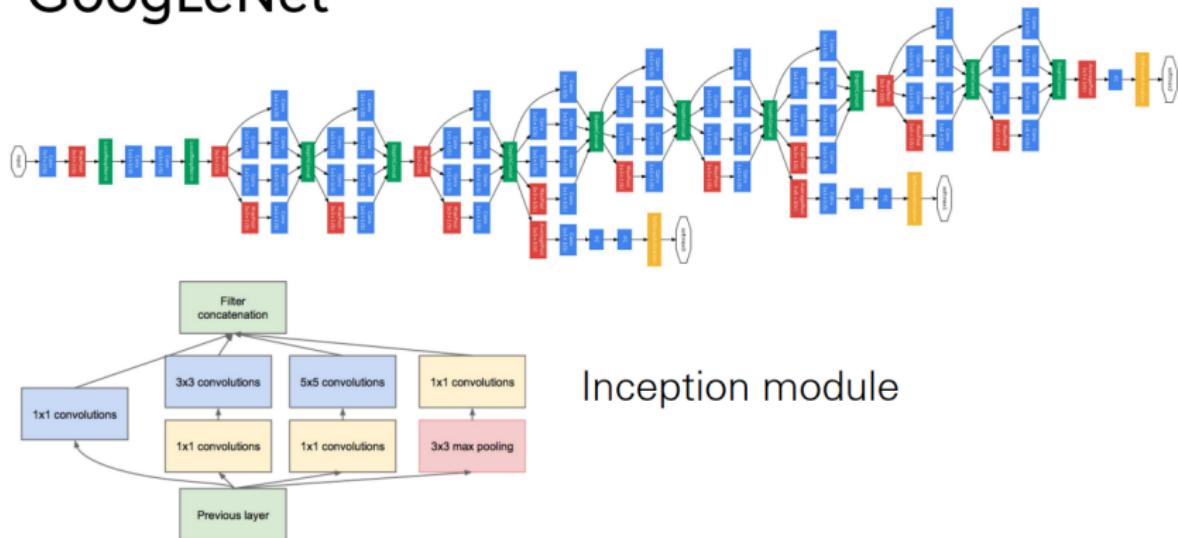


- Branching
- Bootleneck
- Skip connection

Revolution of Depth

GoogLeNet

[Szegedy et al., 2014]



GoogLeNet

[Szegedy et al., 2014]

type	patch size/ stride	output size	depth	#1x1	#3x3 reduce	#3x3	#5x5 reduce	#5x5	pool proj	params	ops
convolution	7x7/2	112x112x64	1							2.7K	34M
max pool	3x3/2	56x56x64	0								
convolution	3x3/1	56x56x192	2		64	192				112K	360M
max pool	3x3/2	28x28x192	0								
inception (3a)		28x28x256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28x28x480	2	128	128	192	32	96	64	380K	304M
max pool	3x3/2	14x14x480	0								
inception (4a)		14x14x512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14x14x512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14x14x512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14x14x528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14x14x832	2	256	160	320	32	128	128	840K	170M
max pool	3x3/2	7x7x832	0								
inception (5a)		7x7x832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7x7x1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7x7/1	1x1x1024	0								
dropout (40%)		1x1x1024	0								
linear		1x1x1000	1							1000K	1M
softmax		1x1x1000	0								

Fun features:

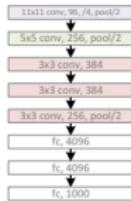
- Only 5 million params!
(Removes FC layers completely)

Compared to AlexNet:

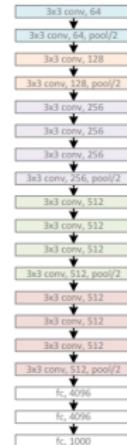
- 12X less params
- 2x more compute
- 6.67% (vs. 16.4%)

Revolution of Depth

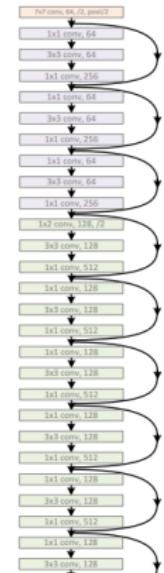
AlexNet, 8 layers
(ILSVRC 2012)



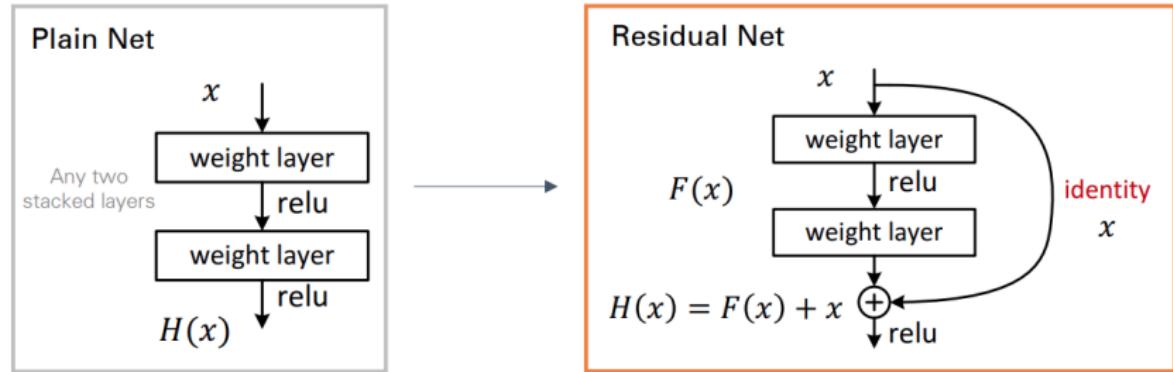
VGG, 19 layers
(ILSVRC 2014)



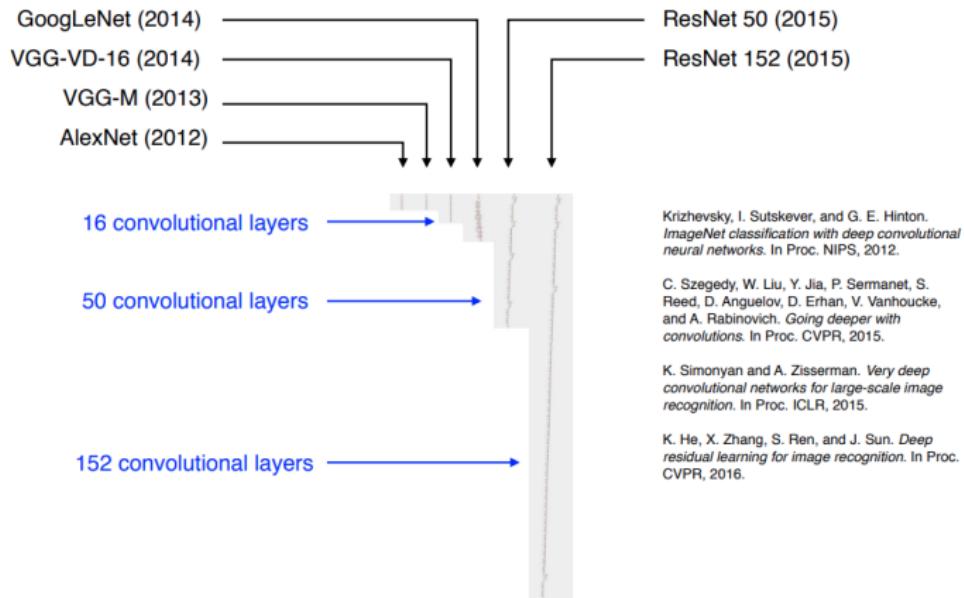
ResNet,
152 layers
(ILSVRC 2015)



Residual Net (ResNet)



How deep is enough?



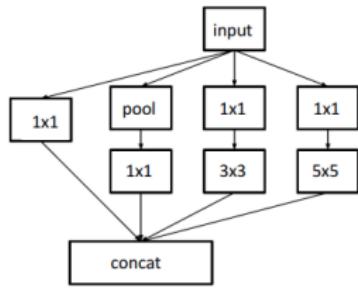
Krizhevsky, I. Sutskever, and G. E. Hinton.
ImageNet classification with deep convolutional neural networks. In Proc. NIPS, 2012.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. *Going deeper with convolutions*. In Proc. CVPR, 2015.

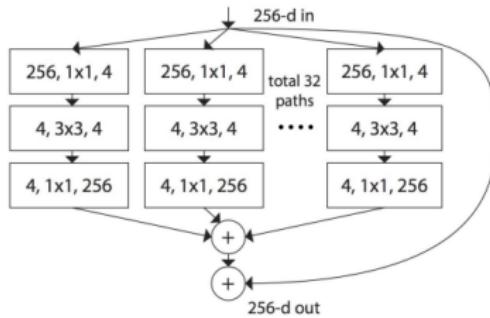
K. Simonyan and A. Zisserman. *Very deep convolutional networks for large-scale image recognition*. In Proc. ICLR, 2015.

K. He, X. Zhang, S. Ren, and J. Sun. *Deep residual learning for image recognition*. In Proc. CVPR, 2016.

ResNeXt: Both Wider and Deeper



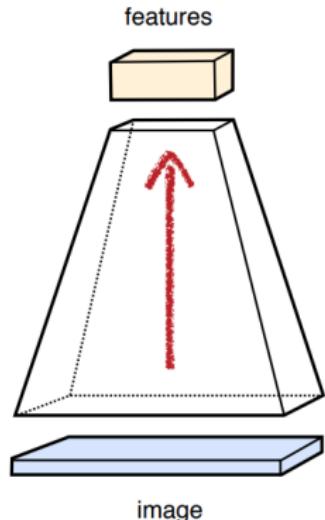
Inception:
heterogeneous multi-branch



ResNeXt:
uniform multi-branch

- shortcut, bottleneck, and multi-branch
- Better accuracy (when having the same FLOPs/params as ResNet)

Design Guidelines



Guideline 1: Avoid tight bottlenecks

- **From bottom to top**

- The spatial resolution $H \times W$ decreases
- The number of channels C increases

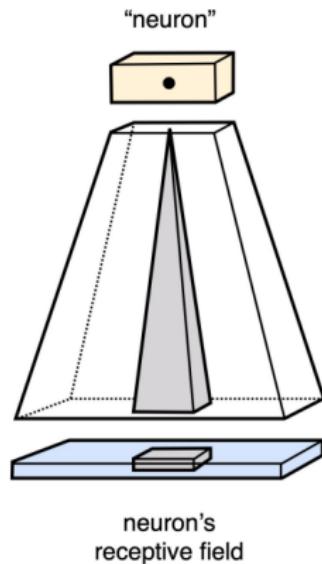
- **Guideline**

- Avoid tight information bottleneck
- Decrease the data volume $H \times W \times C$ slowly

K. Simonyan and A. Zisserman. **Very deep convolutional networks for large-scale image recognition.** In ICLR 2015.

C. Szegedy, V. Vanhoucke, S. Ioffe, and J. Shlens. **Rethinking the inception architecture for computer vision.** In CVPR 2016.

Receptive Field



Must be large enough

- **Receptive field of a neuron**

- The image region influencing a neuron
- Anything happening outside is invisible to the neuron

- **Importance**

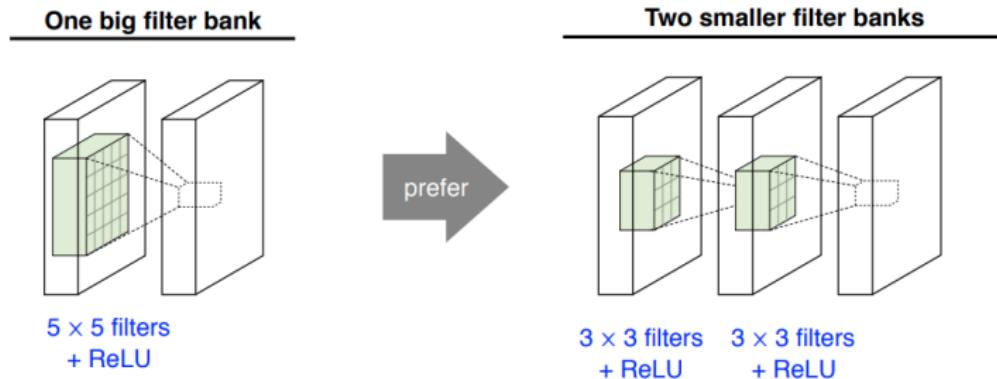
- Large image structures cannot be detected by neurons with small receptive fields

- **Enlarging the receptive field**

- Large filters
- Chains of small filters

Design Guidelines

Guideline 2: Prefer small filter chains



- **Remark:** 101 ResNet layers same size/speed as 16 VGG-VD layers
- **Reason:** Far fewer feature channels (quadratic speed/space gain)
- **Moral:** Optimize your architecture

Transfer Learning with Deep Networks

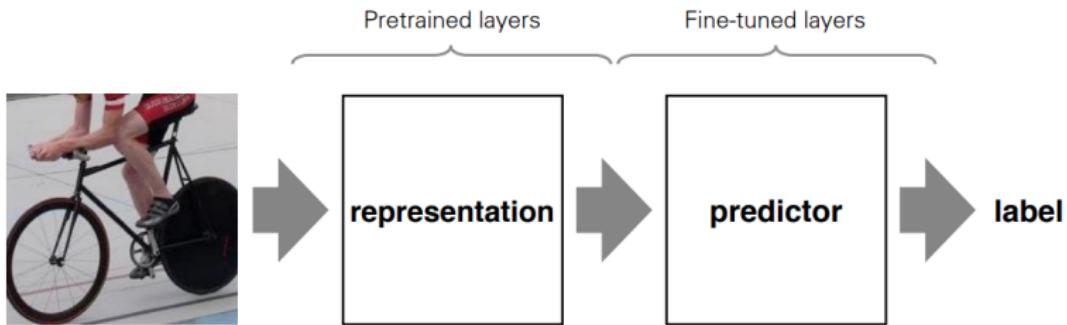
“You need a lot of data if you want to train/use CNNs”

Pre-training and Transfer Learning

“You need a lot of data if you want to train/use CNNs”

BUSTED

Image Representations



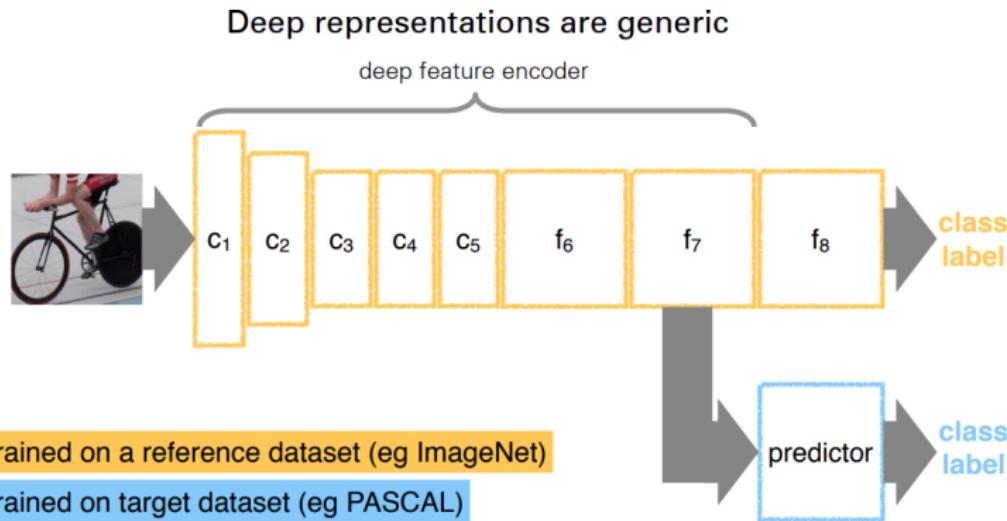
CNN as universal representations

- First several layers in most CNNs are generic
- They can be reused when training data is comparatively scarce.

Application

- Pre-train on ImageNet classification 1M images
- Cut at some deep conv or FC layer to get features

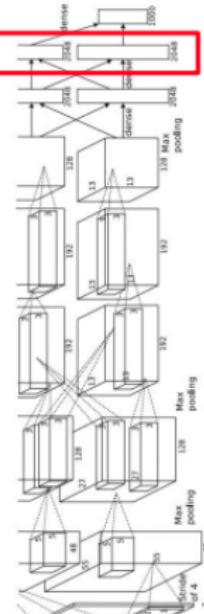
Transfer Learning



- A general purpose deep encoder is obtained by chopping off the last layers of a CNN trained on a large dataset.

Transfer Learning with CNNs

Transfer Learning with CNNs



Test image

L2 Nearest neighbors in feature space

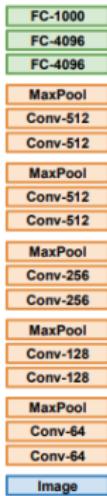


More on this later

Transfer Learning with CNNs

Transfer Learning with CNNs

1. Train on Imagenet

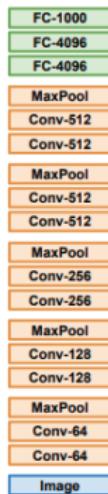


Ionanue et al., "DECAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014
Razavian et al., "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition", CVPR Workshops 2014

Transfer Learning with CNNs

Transfer Learning with CNNs

1. Train on Imagenet



2. Small Dataset (C classes)

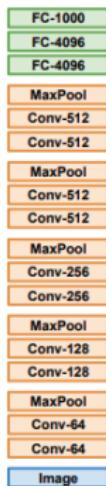


Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014
Razavian et al, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition", CVPR Workshops 2014

Transfer Learning with CNNs

Transfer Learning with CNNs

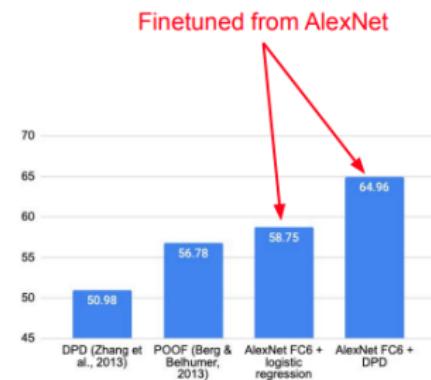
1. Train on Imagenet



2. Small Dataset (C classes)



Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014
Razavian et al, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition", CVPR Workshops 2014

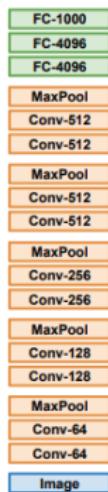


Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014

Transfer Learning with CNNs

Transfer Learning with CNNs

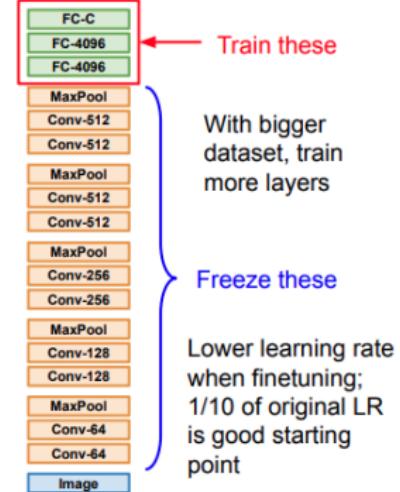
1. Train on Imagenet



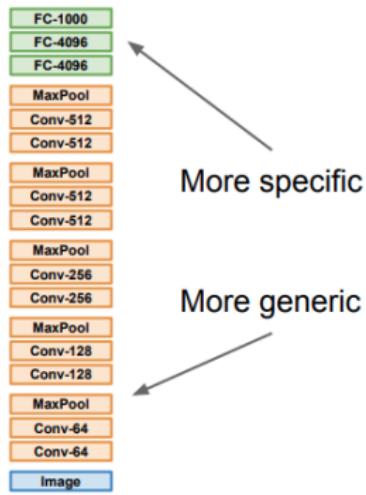
2. Small Dataset (C classes)



3. Bigger dataset

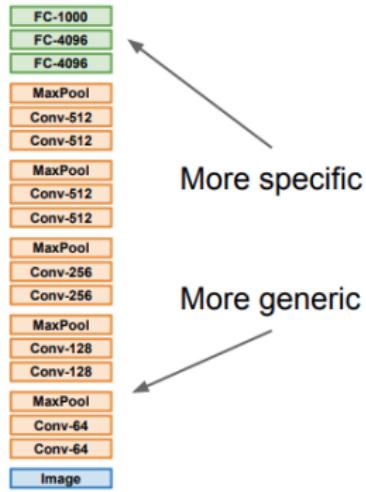


Transfer Learning with CNNs



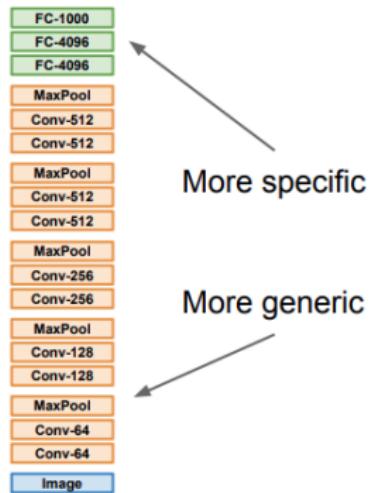
	very similar dataset	very different dataset
very little data	?	?
quite a lot of data	?	?

Transfer Learning with CNNs



	very similar dataset	very different dataset
very little data	Use Linear Classifier on top layer	?
quite a lot of data	Finetune a few layers	?

Transfer Learning with CNNs



	very similar dataset	very different dataset
very little data	Use Linear Classifier on top layer	You're in trouble... Try linear classifier from different stages
quite a lot of data	Finetune a few layers	Finetune a larger number of layers

How to deal the temporal things in DL

Next Class..